COMPLEXITY LEVEL ONE

I. Initialization

1. Populate the network on node at a time. To do this:
    a. Randomize gender based on gender probabilities
    b. Once a gender is selected, discern if its male or female and add the node to the appropriate gender list.
    c. Subsequently, add the node to the network with the following node attributes:
        i. Randomized gender selected
        ii. Randomized physical attractiveness rating from 0 to 1.
    d. Initialized edge parameters that each node possesses including: message, like, dislike, match, and like score.
2. Initialized the counts for the number of matches, male_likes, and female_likes.
3. Created the empty list for the like record, dislike record, and match records.
4. Initialize the like score records for males and females separately to be utilized later in the code.
5. Define the variables for the different node attributes so that we can find characteristics of specific nodes later on.

II.Node Interactions

1. The network evolves through a double nested for loop, the first loop is the amount of days and the second loop is the amount of avg swipes per person. It equates to the amount of swipes per person per day.
2. Grab a random node and label it node_a
3. Find all the node attributes for node_a.
4. Grab an individual that fits the criteria of a potential match
    a. Has to be the opposite gender and an age within the age range that was specified earlier in the code
    b. They also should not have interacted with each other before (like/dislike)
5. Find all the node attributes for node_b
6. Retrieve like scores and like lists for node_a to node_b and node_b to node_a, where they vary based on gender. The like lists are just lists that are appended to keep each like score that is created for percentiles later in the code.
    a. Gender
        i. Males
            1. Attractiveness: 40%
            2. Age Similarity: 10%

            3. Attractiveness Similarity: 20%
            4. Age Difference: 30%
        ii. Females
            1. Attractiveness: 30%
            2. Age Similarity: 20%
            3. Attractiveness Similarity: 10%
            4. Age Difference: 40%

7. The nodes then decide on an action. Here, they use the like score to determine if the probability that they like to be higher or lower. This also differs between males and females.
    a. Males:
        i. If the like score is higher than 0.5, then the chance that node_a will like node_b is 80%
        ii. If the like score is less than 0.5, then the chance that node_a will like node_b is 30%
        iii. If the decide not to like the individual, then they will dislike them
    b. Females:
        i. If the like score is higher than 0.7, then the chance that node_a will like node_b is 80%
        ii. If the like score is less than 0.7, then the chance that node_a will like node_b is 30%
        iii. If the decide not to like the individual, then they will dislike them
8. After the agent makes that action, the code checks to see if there is a mutual like between node_a and node_b. If there is, a match is generated.

COMPLEXITY LEVEL TWO

I. Initialization

1. Populate the network on node at a time. To do this:
    a. Randomize gender based on gender probabilities
    b. Once a gender is selected, discern if its male or female and add the node to the appropriate gender list.
    c. Subsequently, add the node to the network with the following node attributes:
        i. Randomized gender selected
        ii. Randomized physical attractiveness rating from 0 to 1.
    d. Initialized edge parameters that each node possesses including: message, like, dislike, match, and like score.
2. Initialized the counts for the number of matches, male_likes, and female_likes.
3. Created the empty list for the like record, dislike record, and match records.
4. Initialize the like score records for males and females separately to be utilized later in the code.
5. Define the variables for the different node attributes so that we can find characteristics of specific nodes later on.

II.Node Interactions

1. The network evolves through a double nested for loop, the first loop is the amount of days and the second loop is the amount of avg swipes per person. It equates to the amount of swipes per person per day.
2. Grab a random node and label it node_a
3. Find all the node attributes for node_a.
4. Grab an individual that fits the criteria of a potential match
    a. Has to be of the opposite gender and possess an age within the age range that was specified earlier in the code
    b. They also should not have interacted with each other before (like/like with message/dislike)
5. Find all the node attributes for node_b
6. Retrieve like scores and like lists for node_a to node_b and node_b to node_a, where they vary based on gender. The like lists are just lists that are appended to keep each like score that is created for percentiles later in the code.
    a. Gender
        i. Males
            1. Attractiveness: 40%
            2. Age Similarity: 10%

            3. Attractiveness Similarity: 20%
            4. Age Difference: 30%
        ii. Females
            1. Attractiveness: 30%
            2. Age Similarity: 20%
            3. Attractiveness Similarity: 10%
            4. Age Difference: 40%

7. The nodes then decide on an action. First, a check occurs to see if node_b already liked node a. If this is the case, the probability of liking increases.
8. Subsequently, the like score of node a to b is assessed to determine if the probability that they like or like with message to be higher or lower. This also differs between males and females.
    a. Males:
        i. If the like score is higher than 0.5, then the chance that node_a will like node_b is 60%
        ii. If the like score is higher than 0.5, then the chance that node_a will like with message node_b is 20%
        iii. If the like score is less than 0.5, then the chance that node_a will like with message node_b is 5%. The chance of liking is between 5% and 10%.
        iv. If the decide not to like them or dislike the individual, then they will dislike them
    b. Females:
        i. If the like score is higher than 0.6, then the chance that node_a will like node_b is 60%
        ii. If the like score is higher than 0.5, then the chance that node_a will like with message node_b is 20%
        iii. If the like score is less than 0.5, then the chance that node_a will like with message node_b is 3%. The chance of liking is between 3% and 6%.
        iv. If the decide not to like them or dislike the individual, then they will dislike them
9. After the agent makes that action, the code checks to see if there is a mutual like between node_a and node_b. If there is, a match is generated.
10. Subsequently, the messages are generated.

COMPLEXITY LEVEL THREE

I. Initialization

1. Populate the network on node at a time. To do this:
    a. Randomize gender based on gender probabilities
    b. Once a gender is selected, discern if its male or female and add the node to the appropriate gender list.
    c. Subsequently, add the node to the network with the following node attributes:
        i. Randomized gender selected
        ii. Randomized physical attractiveness rating from 0 to 1
        iii. Randomized age between the values of 19 to ages (which is set in the starting block of initialization the network).
        iv. Set amount of coins to start with for each node (denoted as starting_coins in the python source file).
2. Initialized edge parameters that each node possesses including: message, like, dislike, match, and like score.
3. Initialized the counts for the number of matches, male_likes, and female_likes.
4. Created the empty list for the like record, dislike record, and match records.
5. Initialize the like score records for males and females separately to be utilized later in the code.
6. Define the variables for the different node attributes so that we can find characteristics of specific nodes later on.

II. Node Interactions

1. Initialize Male Swipes. To do this:
    a. Select a random male node from the male_list. Call the node, node_a.
    b. Find all node attributes of node_a including: age, attractiveness, ethnicity, education, profession, and coins.
    c. Determine which action node_a decides. If node_a selects "Swipe" (85% of the time), then:
        i. Use the potential_swipe_for_male function to generate potential female match denoted as node_b.

        ii. Find node attributes for node_b.
        iii. Calculate the like score of node_a to node_b and node_b to node_a based on the weights of the different node attributes.

        iv. After the like scores are calculated,use the Action2 function to determine whether a like or dislike occurs from node_a to node_b. Change the edge attribute respectively.

   d. If node_a returns the "Browse" action (15% of the time when node_a has enough coins), then:
        I. Use the Search function to generate node b.
   e. If node_a likes node_b and node_b likes node_a, generate a match. Increment the match counter accordingly and set the "match" edge attribute to 1.

2. Initialize Female Swipes. To do this.
   a. Select a random male node from the female_list. Call the node, node_a.
   b. Find all node attributes of node_a including: age, attractiveness, ethnicity, education, profession, and coins.
   c. Determine which action node_a decides. If node_a selects "Swipe", then:
      i. Use the potential_swipe_for_female function to generate potential female match denoted as node_b.
      ii. Find node attributes for node_b.
      iii. Calculate the like score of node_a to node_b and node_b to node_a based on the weights of the different node attributes.
      iv. Subsequently, use the Action2 function to determine whether a like or dislike occurs from node_a to node_b. Change the edge attribute respectively.
   d. Else, if node_a returns the "Browse" action, then:
      i. Use the Search function to generate node b
   e. If node_a likes node_b and node_b likes node_a, generate a match. Increment the match counter accordingly and set the "match" edge attribute to 1.
3. Generate Messages. To do this, use the message function:

The message function works as follows:

1. Retrieve tbe list_score_m, the like scores generated for male nodes. Find the 33rd and 66th percentile of list of like scores for males. Any value above the 66th percentile is high. Any value below the 33rd percentile is low. Any value in between is “mid”. The probability that a male will message first is 0.206.

2. Retrieve tbe list_score_f, the like scores generated for female nodes. Find the 33rd and 66th percentile of list of like scores for females. Any value above the 66th percentile is high. Any value below the 33rd percentile is low. Any value in between is “mid”. The probability that a male will message first is 0.251.
3. Iterate using a for loop a number of 3* number of matches, generating a random node pair each time.

4. Select a random match pair from match records. Check if match pair selected is not already in message list. If this is the case,
    a. Let the first node in match pair be denoted as node a. Let the second node in match pair be denoted as node b. Retrieve the genders of node a and node b.
5. If node a is male, probability it is the messenger is 73% . If node a is female, probability it is the messenger is 27% .

----------Next Step-----------

6. If match pair is already in the message list, the messenger is the 1st element, and receiver is 2nd element.
7. Retrieve their genders. Find the values of their like scores to one another. If the gender of receiver is male and the messenger’s like score is below the 33rd percentile, decrease the probability of the male messaging. Else, if its in between the 33rd and 66th percentile, then the probability of male messaging remains the same. Else, if it is higher than the 66th percentile, increase the probability of male messaging.
8. Subsequently, generate a random number between 0 and 1. If the number generated is below the probability of male messaging, generate a message. Append the message list. Alter the message edge attribute accordingly.

9. Repeat the same actions for females.

FUNCTION DEFINITION:

The potential swipe function for males works as follows:

1. If the gender of node a is male, find a list of all potential nodes in the network that are within the age range of node_a (Set during initialization of model).
2. Randomly select a female node from potential nodes.
3. If node_a has already liked or disliked node_b, randomly select another female node from potential node. Else, if node_a has not like node_b prior, return node_b.

FUNCTION DEFINITION:

The like_score function behaves as follows:

Assumptions:

1. Males care more about attractiveness than females.
2. Females value finding a male node that is older. Males value finding a female that is younger than them however, they do not believe fulfilling age preferences are as important as females.
3. Males and Females value finding a potential match that is within their age range equally.
4. Males value attractiveness more than females.
5. Women possess a stronger preference for finding a match that is of their own preference. Men do not value it as much.
6. Women prefer men with higher education than they possess. Men prefer women lower education status than them. However, the women values the fulfillment of their education preference more than men.
7. Women care about finding a potential match that is of higher or equal profession. Men do not value it at all.

FUNCTION DEFINITION:

The Action2 function works as follows

1. First, check if node_b has:
   a. liked with message node_a by checking the mess_like_record. If so, increase the probability of node_a liking node_b by 0.1 .
   b. If node_b liked node_a without a message, increase the probability of node_a liking node_by by 0.05 .
   c. Else, if none of the following conditions occurred, probabilities remain the same.
2. Check gender of node a.
   a. If node a is male, generate a random float number between 0 and 1. If node_a likes node_b more than fifty percent, then node_a will like node_b 60% of the time unless node b already liked node a. In such a case, node a will like node b 65% or 70% of the time.
   b. If the random number generated is between 0.6 to 0.8, then node_a decides to like node b with a message (set at initialization).
   c. Else, node_a dislike nodes_a 20% of the time.
   d. If node_a likes node_b less than 50%, then the probability of node_a liking node_b is reduced to 5%. If the number generated is between 5% to 10%, then a like with message occurs from node to node_b.
3. If node b is female:
   a. generate a random float number between 0 and 1. If node_a likes node_b more than sixty percent, then node_a will like node_b 60% of the time unless node b already liked node a. In such a case, node a will like node b 65% or 70% of the time.
   b. If the random number generated is between 0.6 to 0.8, then node_a decides to like node b with a message. Reduce the number of coins node a has by the cost (set at initialization).
   c. Else, node_a dislike nodes_a 20% of the time.
   d. If node_a likes node_b less than 50%, then the probability of node_a liking node_b is reduced to 4%. If the number generated is between 3% to 6%, then a like with message occurs from node to node_b.
   e. Append appropriate record and edge attribute.
4. Return the different records, and counts, and coins of node a, etc.

FUNCTION DEFINITION:

The search function works as follows:

1. Determine gender of node_a. If node a is male, then:
   a. Generate a random node_b using potential_swipe_for_male function.
   b. Find all node attributes of node b.
   c. Calculate the like score of node_a to node_b
   d. If like_a (towards node_b) is higher than 0.7, then 80% of the time, a like is generated from node a to node b. Subsequently, the number of coins node a has drops by cost of such a like. Append to like record.
2. If gender_a is female, then use the potential swipe function of the first complexity to generate a node b.
   a. Find all node attributes of node_b.
   b. Calculate like score from node_a to node_a.
   c. If like_a is greater than 0.8, then node_a will like node_b, 80% of the time. Coins will be deducted and the like will be recorded.

FUNCTION DEFINITION:

The potential swipe function for females works as follows:

1. Find all the male nodes that have liked node_a (female).
2. Randomly select one of the male nodes as a potential match.
3. If node_a has already liked or disliked node b, then find another potential male candidate. Else, return node b (male).